3

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-268896
(43)Date of publication of application : 20.09.2002

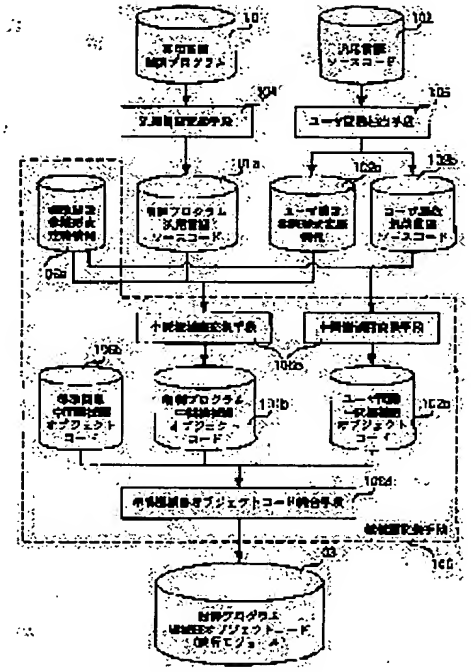
(51)Int.Cl. G06F 9/45
G05B 19/05

(21)Application number : 2001-068674
(22)Date of filing : 12.03.2001

(71)Applicant : HITACHI LTD
(72)Inventor : KOYAMA MASAHIRO
HOSODA YUJI
MATSUSHITA TSURUMASA
SHIMOZU TADAO

(54) METHOD AND DEVICE FOR GENERATING CONTROL PROGRAM

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a method for generating a control program by which wasteful labor and time for quoting a user function is relieved and an input mistake is reduced.
SOLUTION: A user function extracting means 105 generates the reference form definition information 102a of the user function to be utilized in the control program and a user function general purpose language source code 102b from a source code 102 which is described by a user through the use of a general purpose language. A general purpose language converting means 104 converts the control program 101 where the description of the user function is mixed into a control program general purpose language source code 101a and a machine language converting means 106 converts the source code 101a into a machine language through the use of definition information 102a and the source code 102b.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－268896
（P2002－268896A）
(43)公開日　平成14年９月20日(2002.9.20)

(51)Int.Cl.'	識別記号	ＦＩ		テーマコード'（参考）
Ｇ０６Ｆ　9/45		Ｇ０６Ｆ　9/44	３２２Ｍ	５Ｂ０８１
Ｇ０５Ｂ　19/05		Ｇ０５Ｂ　19/05	Ａ	５Ｈ２２０

審査請求　未請求　請求項の数５　ＯＬ　（全 13 頁）

(21)出願番号　特願2001－68674(P2001－68674)

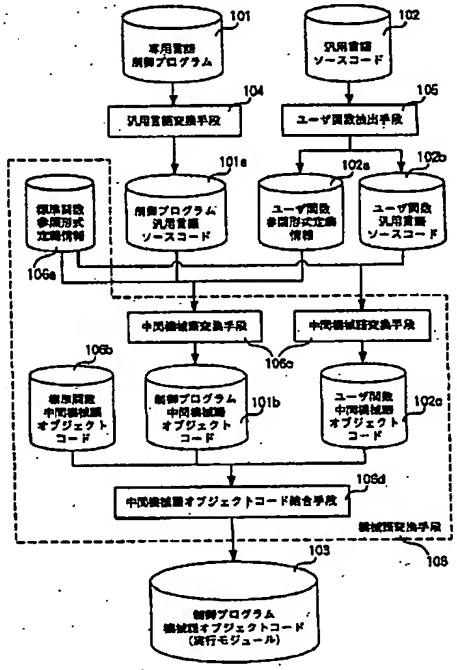
(22)出願日　平成13年３月12日(2001.3.12)

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　小山　昌宏
茨城県土浦市神立町502番地　株式会社日立製作所機械研究所内
(72)発明者　細田　祐司
茨城県土浦市神立町502番地　株式会社日立製作所機械研究所内
(74)代理人　100093872
弁理士　高崎　芳紘

最終頁に続く

(54)【発明の名称】　制御プログラム作成方法とその装置

(57)【要約】
【課題】　ユーザ関数を引用するための余計な手間と入力ミスを削減できる制御プログラムの作成方法を提供する。
【解決手段】　ユーザ関数抽出手段１０５は、ユーザが汎用言語で記述したソースコード１０２から、制御プログラムの中で利用するユーザ関数の参照形式定義情報１０２ａとユーザ関数汎用言語ソースコード１０２ｂを生成する。汎用言語変換手段１０４は、ユーザ関数の記述が混在した制御プログラム１０１を制御プログラム汎用言語ソースコード１０１ａに変換し、機械語変換手段１０６は、定義情報１０２ａとソースコード１０２ｂを用いて、ソースコード１０１ａを機械語に変換する。



1

【特許請求の範囲】

【請求項１】　問題向きの専用言語により記述された制御プログラムを制御装置が実行可能な機械語に変換するための制御プログラム作成装置において、

ユーザが汎用言語により記述したソースコード群から、制御プログラムの中で利用するユーザ関数の参照形式定義情報と、前記ユーザ関数に関連する処理のみを抽出したユーザ関数汎用言語ソースコードとを生成するユーザ関数抽出手段と、

専用言語の記述の中に前記ユーザ関数の記述を混在させた制御プログラムから制御プログラムの汎用言語ソースコードを生成する汎用言語変換手段と、

前記ユーザ関数抽出手段により生成された参照形式定義情報及びユーザ関数汎用言語ソースコードを用いて、前記汎用言語変換手段により生成された制御プログラムの汎用言語ソースコードを前記機械語に変換する機械語変換手段と、

を備えたことを特徴とする制御プログラム作成装置。

【請求項２】　前記ユーザ関数抽出手段は、前記汎用言語により記述したソースコード群に含まれる関数の宣言文とこれらの関数対応のチェックボックスを持つチェックリストを生成、表示するチェックリスト表示手段を備え、チェックボックスにより選択されたユーザ関数に対して前記参照形式定義情報と前記ユーザ関数汎用言語ソースコードとを自動的に生成することを特徴とする請求項１記載の制御プログラムの作成装置。

【請求項３】　前記専用言語は、ユーザ関数を利用して汎用言語により記述されたブロックを指示するための指示コマンドを有しており、前記汎用言語変換手段は、前記制御プログラムの、前記指示コマンドで指定されていない専用言語のコマンドを汎用言語の標準関数に変換し、前記指示コマンドで指示されたブロックの内容をそのままコピーすることにより制御プログラムの汎用言語ソースコードを生成することを特徴とする請求項１記載の制御プログラム作成装置。

【請求項４】　前記制御プログラムは、シーケンシャル・ファンクション・チャートの図形形式の記述とテキスト形式の記述を組み合わせて作成したプログラムを、シーケンスの各ステップの状態遷移に対応したＩＦ～ＴＨＥＮ形式のルールによる記述に変換して保存されたプログラムであり、前記汎用言語変換手段は、前記ルールの各々をステップが活性化されたときに実行される関数に変換して制御プログラムの汎用言語ソースコードを生成することを特徴とする請求項１記載の制御プログラム作成装置。

【請求項５】　問題向きの専用言語により記述された制御プログラムを制御装置が実行可能な機械語に変換するための制御プログラム作成方法において、

ユーザ関数を利用して汎用言語により記述されたブロックを、そのブロックを指示するための指示コマンドを用いて制御プログラム中に記述し、

前記ユーザ関数の各々のソースコードから当該ユーザ関数の参照定義情報と当該ユーザ関数の処理を記述するユーザ関数汎用言語ソースコードとを生成し、

前記制御プログラムの、前記指示コマンドで指示されていない専用言語の記述を制御プログラム汎用言語ソースコードに変換した後、この制御プログラム汎用言語ソースコードを前記参照定義情報及び前記ユーザ関数汎用言語ソースコードを用いて機械語に変換することを特徴とする制御プログラム作成方法。

2

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、問題向きの専用言語により記述した制御プログラムを翻訳して制御装置が実行可能な機械語を生成する制御プログラム作成方法とその装置に関する。

【０００２】

【従来の技術】ロボット、プラント等の制御装置では、問題向きの専用言語（例えば、ロボット言語やラダー図など）により制御プログラムを記述し、この制御プログラムをコンパイラにより機械語に翻訳したのちに実行するか、あるいはこの制御プログラムをインタプリタにより逐次解釈及び実行する。

【０００３】このような制御装置では、ユーザが容易に制御プログラムを作成できるように、各種制御機能を実現する標準関数ライブラリが用意されており、これを専用言語の命令として利用できる。また、この標準関数に含まれない制御機能をユーザが必要とする場合には、予め用意された専用言語の命令を組み合わせてその機能を実現するか、あるいは何らかの方法で標準関数ライブラリにユーザの仕様に基づいた新しい関数（ユーザ関数）を追加して、これを専用言語の新しい命令として利用できるようにコンパイラあるいはインタプリタを改造する必要がある。

【０００４】しかし、前者の専用言語の命令を組み合わせた記述による場合には、その記述能力に限界があり、ユーザの必要とする制御機能を記述できない可能性がある。あるいは、仮に記述できたとしても、記述が複雑になってしまい、プログラムとして解読性が低下する可能性がある。また、インタプリタによる実行の場合、このように専用言語により複雑な演算処理を記述すると、プログラムの解釈処理時間が増大して、制御システムの応答性が低下する可能性がある。一方、ユーザ関数をライブラリへ追加するためには、一般的にはメーカにソフトウェアの改造を依頼することになり、このために多額の費用が発生する可能性がある。また、ユーザが必要とする制御機能をユーザ自身が汎用言語によるプログラムとして所有している場合もあり、このような既存のソフトウェア資産を有効活用したいという要望もあったが、従来の制御装置ではこれを容易に実現できなかった。

3

【０００５】このような問題点を解決する試みに関する公知例としては、特開平１１－３３８５２１号公報に記載された制御装置が挙げられる。この公知例では、制御装置の基本ソフトウェアにユーザ関数登録機能を備える発明が述べられている。このユーザ関数登録機能では、ユーザがユーザ関数の名称と引数等を定義する文法、ユーザ関数の演算内容を汎用言語により記述したもの、さらにユーザ関数の実行モジュールのメモリ上でのエントリアドレスを入力することにより、専用言語の標準関数とは別のユーザ関数を登録できるようになっている。この公知技術によれば、制御プログラムの作成時にはユーザ関数をそのまま記述できるので、プログラムの解読性が低下することはなく、また制御プログラムの実行時にはユーザ関数の実行コードを呼び出すので、その解釈処理時間が大幅に増大することはないと考えられる。また、制御装置の改造を第三者に依頼する必要はなく、ユーザ自身が所望のユーザ関数を新たに登録することができる。

【０００６】

【発明が解決しようとする課題】上記した特開平１１－３３８５２１号公報に記載されたユーザ関数登録機能では、ユーザ自身がユーザ関数の名称と引数等を定義する文法とユーザ関数の実行モジュールのエントリアドレス等を所定の書式にしたがって入力する必要があるため、ユーザに余計な手間をかけてしまうだけでなく、この段階で入力ミスが発生する可能性もあり、プログラム作成上の効率の面でさらなる改善が望まれる。また、ユーザが既に汎用言語によるソフトウェア資産を持っており、これを部分的に利用したいと考えている場合、上記ユーザ関数登録機能では、そのソースコードを再編集する必要が生じる可能性がある。さらに、上記公知例では、制御プログラムをインタプリタ方式により実行することを前提としているため、プログラム実行効率の面でも改善の余地はある。

【０００７】そこで、本発明では、ユーザが汎用言語で作成した関数を、簡単な操作で制御プログラムに組み込んで利用でき、かつその組み込みにより実行速度が特に低下しない制御プログラム作成方法とその装置を提供することを目的とする。

【０００８】

【課題を解決するための手段】本発明は、問題向きの専用言語により記述された制御プログラムを制御装置が実行可能な機械語に変換するための制御プログラム作成装置において、ユーザが汎用言語により記述したソースコード群から、制御プログラムの中で利用するユーザ関数の参照形式定義情報と、前記ユーザ関数に関連する処理のみを抽出したユーザ関数汎用言語ソースコードとを生成するユーザ関数抽出手段と、専用言語の記述の中に前記ユーザ関数の記述を混在させた制御プログラムから制御プログラムの汎用言語ソースコードを生成する汎用言語変換手段と、前記ユーザ関数抽出手段により生成された参照形式定義情報及びユーザ関数汎用言語ソースコードを用いて、前記汎用言語変換手段により生成された制御プログラムの汎用言語ソースコードを前記機械語に変換する機械語変換手段と、を備えたことを特徴とする制御プログラム作成装置を開示する。

4

【０００９】更に本発明は、前記ユーザ関数抽出手段は、前記汎用言語により記述したソースコード群に含まれる関数の宣言文とこれらの関数対応のチェックボックスを持つチェックリストを生成、表示するチェックリスト表示手段を備え、チェックボックスにより選択されたユーザ関数に対して前記参照形式定義情報と前記ユーザ関数汎用言語ソースコードとを自動的に生成することを特徴とする制御プログラムの作成装置を開示する。

【００１０】更に本発明は、前記専用言語は、ユーザ関数を利用して汎用言語により記述されたブロックを指示するための指示コマンドを有しており、前記汎用言語変換手段は、前記制御プログラムの、前記指示コマンドで指定されていない専用言語のコマンドを汎用言語の標準関数に変換し、前記指示コマンドで指示されたブロックの内容をそのままコピーすることにより制御プログラムの汎用言語ソースコードを生成することを特徴とする制御プログラム作成装置を開示する。

【００１１】更に本発明は、前記プログラムは、シーケンシャル・ファンクション・チャートの図形形式の記述とテキスト形式の記述を組み合わせて作成プログラムを、シーケンスの各ステップの状態遷移に対応したＩＦ～ＴＨＥＮ形式のルールによる記述に変換して保存されたプログラムであり、前記汎用言語変換手段は、前記ルールの各々をステップが活性化されたときに実行される関数に変換して制御プログラムの汎用言語ソースコードを生成することを特徴とする制御プログラム作成装置を開示する。

【００１２】更に本発明は、問題向きの専用言語により記述された制御プログラムを制御装置が実行可能な機械語に変換するための制御プログラム作成方法において、ユーザ関数を利用して汎用言語により記述されたブロックを、そのブロックを指示するための指示コマンドを用いて制御プログラム中に記述し、前記ユーザ関数の各々のソースコードから当該ユーザ関数の参照定義情報と当該ユーザ関数の処理を記述するユーザ関数汎用言語ソースコードとを生成し、前記制御プログラムの、前記指示コマンドで指示されていない専用言語の記述を制御プログラム汎用言語ソースコードに変換した後、この制御プログラム汎用言語ソースコードを前記参照定義情報及び前記ユーザ関数汎用言語ソースコードを用いて機械語に変換することを特徴とする制御プログラム作成方法を開示する。

【００１３】

【発明の実施の形態】以下、本発明の実施の形態につい

て、添付図面を参照して説明する。図１は、本発明になる制御プログラム作成装置の処理手順例を示す図である。この制御プログラム作成装置は、プログラマブルロジックコントローラ用のシーケンス制御言語、モーションコントローラ用のモーション制御言語、ロボットコントローラ用のロボット言語などの専用言語に用意された標準関数ライブラリを用いて記述された専用言語プログラム１０１中に、ユーザの作成した汎用言語ソースコード１０２中に含まれるユーザ関数を組み込み、それらを最終的に機械語に翻訳して制御プログラム機械語オブジェクトコード１０３を生成するものである。

【００１４】このために、図１の制御プログラム作成装置には、汎用言語ソースコード１０２から、制御プログラムの中でユーザ関数を利用するためのユーザ関数参照形式定義情報１０２ａと、ユーザ関数に関連する処理のみを抽出したユーザ関数汎用言語ソースコード１０２ｂとを自動的に生成するユーザ関数抽出手段１０５を設ける。ここで抽出されたユーザ関数は、専用言語制御プログラム１０１の中で記述することが可能となる。このように専用言語の記述の中にユーザ関数の記述を混在させた専用言語制御プログラム１０１は、汎用言語変換手段１０４によって制御プログラム汎用言語ソースコード１０１ａに変換される。機械語変換手段１０６は、以上のように生成された制御プログラム汎用言語ソースコード１０１ａ、ユーザ関数参照形式定義情報１０２ａ、及びユーザ関数汎用言語ソースコード１０２ｂから、最終的に制御装置内で実行可能な形式の制御プログラム機械語オブジェクトコード１０３を生成する。以下では、汎用言語としては一般に広く普及しているＣ言語を用いるものとする。

【００１５】機械語変換手段１０６では、制御プログラム汎用言語ソースコード１０１ａ、ユーザ関数汎用言語ソースコード１０２ｂは、中間機械語変換手段１０６ｃによって、それぞれ制御プログラム中間機械語オブジェクトコード１０１ｂ、ユーザ関数中間機械語オブジェクトコード１０２ｃに変換される。ここで、制御プログラム汎用言語ソースコード１０１ａにはユーザ関数が記述されており、このユーザ関数の実体は別のソースコードであるユーザ関数汎用言語ソースコード１０２ｂに記述されている。このようなユーザ関数の外部参照を可能とするためにユーザ関数参照形式定義情報１０２ａが必要となり、制御プログラム汎用言語ソースコード１０１ａを中間機械語に変換する際に用いられる。これと同様に、専用言語及び汎用言語に含まれる各種標準関数の実体は、汎用言語処理システムに備えられている標準関数ライブラリに含まれており、これらの標準関数を外部参照するために標準関数参照形式定義情報１０６ａが必要であり、制御プログラム汎用言語ソースコード１０１ａ、ユーザ関数汎用言語ソースコード１０２ｂを中間機械語に変換する際に用いられる。

【００１６】中間機械語オブジェクトコード結合手段１０６ｄは、制御プログラム中間機械語オブジェクトコード１０１ｂ、ユーザ関数中間機械語オブジェクトコード１０２ｃ、さらに標準関数ライブラリを中間機械語に変換した標準関数中間機械語オブジェクトコード１０６ｂから、標準関数及びユーザ関数の外部参照を解決すると同時にそれぞれの中間機械語オブジェクトコードを結合して、最終的な実行モジュールである制御プログラム機械語オブジェクトコード１０３を生成する。

【００１７】前記のように汎用言語としてＣ言語を用いるものとすると、制御プログラム汎用言語ソースコード１０１ａ、ユーザ関数汎用言語ソースコード１０２ｂは、それぞれＣ言語のソースコードであり、標準関数参照形式定義情報１０６ａ、ユーザ関数参照形式定義情報１０２ａは、それぞれ標準関数、ユーザ関数のＣ言語のｅｘｔｅｒｎ宣言を記述したヘッダファイル（インクルードファイル）に相当する。なお、これらのヘッダファイルは制御プログラム汎用言語ソースコード１０１ａの冒頭部分において，プリプロセッサ命令＃ｉｎｃｌｕｄｅによって取り込まれる。また、中間機械語変換手段１０６ｃ、及び中間機械語オブジェクトコード結合手段１０６ｄは、それぞれＣ（クロス）コンパイラ、及びリンカに相当する。

【００１８】以上の図１に示したプログラム作成装置において、本発明の特徴とするのは、ユーザ関数抽出手段１０５及び汎用言語変換手段１０４であり、以下、これらを中心とした動作を具体例を用いて説明する。

【００１９】図２は、制御プログラムの編集画面の一例を示す図である。ここで、制御プログラムの処理の流れ（シーケンス）はＳＦＣ（シーケンシャル・ファンクション・チャート）により記述し、シーケンスの各ステップの処理はテキスト形式の専用言語により記述する。シーケンス編集ウィンドウ２０１では、制御対象となる各機器のシーケンスをＳＦＣの図形形式により記述する。コマンド編集ウィンドウ２０２では、シーケンスの各ステップの動作を専用言語のコマンドとして記述する。終了条件編集ウィンドウ２０３では、各ステップが終了するための条件を条件式として記述する。シーケンス編集ウィンドウ２０１で記述されるＳＦＣのステップＳ１００、Ｓ１０１…の各々は各機器の単位動作を示し、その動作内容はコマンド列としてコマンド編集ウィンドウ２０２において記述される。

【００２０】図２右側のコマンド編集ウィンドウ２０２には、ステップＳ１０３のコマンド列が記述されており、これとポジショナの目標位置を計算するためのパラメータを取得し、ユーザ関数を用いて目標位置を計算し、さらに、この目標位置へポジショナを移動させ、ワークを把持するためのチャックを開いて、チャック開動作を監視するためのオンディレイタイマを起動するステップである。また、トランジション２０１ｂ、２０１ｃ

は、ステップ間の状態遷移を示し、その遷移条件はこれらのトランジションの入力ステップＳ１０３の終了条件として、終了条件編集ウィンドウ２０３に記述される。ステップＳ１０３の終了条件は２種類あり、制限時間内にチャックが完全に開いた場合（終了条件Ｔ１）、制限時間内にチャックが完全に開かない場合（終了条件Ｔ２）を定義する条件式が記述されている。終了条件Ｔ１、Ｔ２はステップＳ１０３から次のステップに遷移するための条件として、その出力トランジション２０１ｂ、２０１ｃの右横に付記される。

【００２１】コマンド編集ウィンドウ２０２では、制御装置側で予め用意された専用言語のコマンドを記述するだけでなく、ユーザがＣ言語などの汎用言語により作成したユーザ関数を記述することができる。図１の構成では、このようにコマンド列の中にＣ言語によるユーザ関数を記述した部分（以下、ユーザ関数ブロック）を明示するためにコマンド＃Ｃ、＃ＥＮＤ＿Ｃが用意されており、これらで囲まれた部分がユーザ関数ブロックを表している。図２のコマンド編集ウィンドウ２０２に示したコマンド列の中にある＃Ｃと＃ＥＮＤ＿Ｃで囲まれた部分、すなわちユーザ関数ブロック２０２ａには、Ｃ言語で別途定義されたユーザ関数ｃａｌｃ＿ｐｏｓ＿ｘ（）、ｃａｌｃ＿ｐｏｓ＿ｙ（）によりポジショナの目標位置を計算し、これらを専用言語の変数ＶＲ４、ＶＲ５（それぞれＣ言語表記では、ＶＲ〔４〕、ＶＲ〔５〕）に代入する処理が記述されている。変数ＶＲ４、ＶＲ５は、ユーザ関数ブロック２０２ａに続く専用言語のコマンド（ＭＯＶＥ　ＡＸＩＳ１ＶＲ４　ＡＸＩＳ２　ＶＲ５　Ｆ　２０００．）のパラメータとして参照されている。なお、ここに示すように、ユーザ関数ブロック２０２ａの中の表記は、Ｃ言語の文法に従った表記となる。また、専用言語で用意される番号付きの各種変数（ＶＲｎなど）はユーザ関数ブロック２０２ａの中でも利用可能であり、これらはＣ言語の配列（ＶＲ〔ｎ〕など）として表記される。

【００２２】このようなユーザ関数ブロック２０２ａを示すコマンドを設けたことにより、ユーザが慣れ親しんだＣ言語の文法に従って容易にユーザ関数を記述することが可能となる。また、このようにユーザ関数ブロック２０２ａを明示するコマンドを利用すれば、後述のように、専用言語制御プログラム１０１から制御プログラム汎用言語ソースコード１０１ａに変換する汎用言語変換手段１０４の処理を簡単にすることができる。

【００２３】なお、図２のようなＳＦＣでは、並行して実行される各シーケンスの間の同期を表現するためのステップが用意されており、これを同期ステップと呼ぶ。例えば、シーケンス編集ウィンドウ２０１に示した同期ステップＲＥＱ１は、ステップＳ１０３の入力トランジション（２０１ｆ）に対する入力となっており、これは入力トランジション（２０１ｆ）において他のシーケンスが出力する同期信号を待つことを意味する。すなわち、ステップＳ１０３への遷移は、直前のステップＳ１０２が終了条件Ｔ１で終了し、さらに同期ステップＲＥＱ１が活性化されることにより発生する。

【００２４】図４は、ユーザ関数抽出手段１０５の概略処理を示すフローチャート、図３はその処理例を示す説明図である。通常、ユーザに関連するユーザ関数を記述したソースコード群を１つのファイルにまとめているから、この中のいくつかのユーザ関数を利用するときは、まずユーザが該当するファイルを指定して記述されたソースコードをオープンする（処理４０１）。図３のＣ言語のソースコード３０１はその記述例で、ｆｌｏａｔ型の変数ｘ、ｙ、ｃからｆｌｏａｔ型の値を算出する関数ｃａｌｃ＿ｖｅｌ＿ｘ、ｃａｌｃ＿ｖｅｌ＿ｙ、ｃａｌｃ＿ｐｏｓ＿ｘ、ｃａｌｃ＿ｐｏｓ＿ｙ…が記述されている。但し、各関数の処理を定義する｛…｝の部分は、通常複数の行に分けて記述されるが、図３ではこれを｛…｝の一行で表している。次に、表示されたソースコードから、その中のユーザ関数を調べて、ユーザ関数のチェックリスト３０２を作成し、これをユーザに対して提示する（処理４０２）。このチェックリスト３０２は、関数宣言文と各関数に設けたチェックボックスより成っている。ユーザはこのチェックリストを参照し、制御プログラムの中で使用するユーザ関数をチェックボックスをクリックして選択する（処理４０３）。

【００２５】ユーザ関数の選択が終了すれば、選択されたユーザ関数の中で直接的または間接的に呼び出されるすべての関数を調べ（処理４０４）、選択されたユーザ関数とそれに関連する関数のソースコードを抜き出した新たなユーザ関数ソースコード３０４を生成する（処理４０５）。さらに、ユーザ関数のチェックリスト３０２のユーザ関数宣言文から、選択されたユーザ関数の外部参照宣言ヘッダファイル３０３を生成する（処理４０６）。ここで生成されたユーザ関数生成コード３０４及び外部参照ヘッダファイル３０３は、それぞれ図１のユーザ関数汎用言語ソースコード１０２ｂ及びユーザ関数参照形式定義情報１０２ａに相当する。また外部参照ヘッダファイル３０３は、選択されたユーザ関数の関数宣言文に外部参照可能であることを示す“ｅｘｔｅｒｎ”を付加したリストである。

【００２６】以上に述べたユーザ関数抽出手段１０５を利用すれば、ユーザはＣ言語ソースコード３０１から生成されるユーザ関数のチェックリスト３０２にチェックするという簡単な操作を行うだけで選択したユーザ関数を専用言語プログラム中で利用できるようになるので、ユーザ関数を引用するためのユーザの手間と入力ミスを極力省くことができる。また、ユーザが既に汎用言語によるソフトウェア資産を所有している場合、これを容易かつ有効に活用することが可能となる。

【００２７】図６は、汎用言語変換手段１０４の概略処

理を示すフローチャート、図５はその処理例を示す図で、図２に示した専用言語制御プログラム１０１の一部を制御プログラム汎用言語ソースコード１０１ａに変換する場合を示している。図２で説明したように、制御プログラムをＳＦＣで作成するときには、制御プログラムのシーケンスを図２のウィンドウ２０１のようなＳＦＣの図形形式で記述し、シーケンスの各ステップの動作と終了条件を図２のウィンドウ２０２のようにテキスト形式の言語により記述する。このような図形形式とテキスト形式を組み合わせて記述された専用言語制御プログラム１０１は、図５に示すようなテキスト形式の専用言語制御プログラム５１０に自動的に変換され、ファイルとして保存されている。

【００２８】この専用言語制御プログラム５１０はＩＦ～ＴＨＥＮ形式のルールによって記述されており、各ルールの条件部にはシーケンスのあるステップから次のステップへの状態遷移を定義する条件、実行部には遷移した次のステップで実行されるコマンド列が記述される。例えば、ルール５１１の条件部５１１ａには、ステップＳ１０２が正常終了（ＣｏｍｐｌｅｔｅＮｏｒｍａｌｌｙ）し、さらに同期ステップＲＥＱ１が活性化される（ＯＮ）という条件式が記述されている。ルール５１１の実行部の最初のブロック５１１ｂは、ステップの遷移に関するコマンド列であり、ステップＳ１０２と同期ステップＲＥＱ１を不活性状態にし（ＴＥＲＭ　Ｓ１０２、ＴＥＲＭ　ＲＥＱ１）、ステップＳ１０３を実行中状態にする（ＥＸＥＣ　Ｓ１０３）。次に続くブロック５１１ｃ、５１１ｄ、５１１ｅは、図２のコマンド編集ウィンドウ２０２に示したステップＳ１０３のコマンド列であり、ブロック５１１ｃ、５１１ｅは専用言語によるコマンド列、ブロック５１１ｄはＣ言語によるユーザ関数ブロック２０２ａである。最後のＥＮＤ＿ＩＦ文５１１ｆは、ルール５１１の終わりを示す。

【００２９】汎用言語変換手段１０４は、まず、ユーザにより指定された専用言語制御プログラム１０１のファイルをオープンし（処理６０１）、上記のような形式で記述された専用言語制御プログラム５１０を一行ずつ取り込む（処理６０２）。次に取り込んだ行がＩＦ文かを調べ（処理６０３）、ＩＦ文であればこの行から次に現れるＥＮＤ＿ＩＦ文までのルールを１つの関数とすべく、ＩＦ文の第１条件式から関数宣言部（関数の型と名称）を生成し（処理６０４）、さらにこの第１条件式からｉｆ条件文を生成する（処理６０５）。ここで生成されるルール対応の関数は、各ステップが活性化されたときに実行される関数であり、関数名は活性化されるステップ番号に対応している。図５のルール５１１の場合には、図２のステップＳ１０２が活性化（ステップＳ１０２のすべてのコマンド発行が終わっている状態）していることが第１条件式であるので、処理５０４ではルール５１１対応の関数名はステップＳ１０２に対して“ｓｔｔ１０２”とされ、そして処理６０５では、第１条件式から対応するｉｆ条件文“ｉｆ（ｓｔｔ〔１０２〕＝＝Ｃｏｍｐｌｅｔｅ　Ｎｏｒｍａｌｌｙ”が生成される。

【００３０】次の一行を取り込み、これがＩＦ文でなかったときは（処理６０３でＮＯ）、それが＆（アンド）条件であれば（処理６０６でＹＥＳ）、その条件式からｉｆ条件文の続きを生成し（処理６０７）、＆条件でないときは処理６０８へ移行する。図５の例では、２行目はＩＦ文の続きの＆条件“＆　ＲＥＱ１＝ＯＮ”であるから、これに対応するｉｆ条件式“＆＆　Ｒｅｑ〔１〕＝＝ＯＮ”が生成される。処理６０７は＆条件がなくなるまで繰り返される。こうして、ルール５１１の条件部５１１ａは、Ｃ言語のｉｆ文に変換され、ブロック５２１ａが生成される。

【００３１】取り込んだ行がＩＦ文でも＆条件でもないときは（処理６０３、６０６でＮＯ）、その行がＴＨＥＮ文かを判定し（処理６０８）、ＴＨＥＮ文でないときは注釈行等の処理に関係ない文として無視して処理６０２へ戻る。処理６０８の判定結果がＴＨＥＮ文であれば、ｉｆ文の実行部の生成に移る。ここで、さらに１行ずつ取り込み（処理６０９）、これが＃Ｃ文でなく（処理６１０でＮＯ）、ＥＮＤ＿ＩＦ文でもなければ（処理６１１でＮＯ）、これは専用言語のコマンドであり、これをＣ言語の標準関数に変換する（処理６１２）。また、取り込んだ行が「＃Ｃ」であれば（処理６１０でＹＥＳ）、ここからユーザ関数ブロックが始まることになり、さらに１行ずつ取り込んで（処理６１４）、これが＃ＥＮＤ＿Ｃ文でなければ（処理６１５でＮＯ）、取り込んだ文字列、すなわちＣ言語による表記をそのままコピーする（処理６１６）。この処理を＃ＥＮＤ＿Ｃ文に到達するまで（処理６１５でＹＥＳ）繰り返す。このようにユーザ関数ブロックの終わりに到達すれば、再び１行ずつ取り込み（処理６０９）、これがＥＮＤ＿ＩＦ文になるまで、以上のようなｉｆ文の実行部の生成処理を繰り返す（処理６１１でＹＥＳ）。ＥＮＤ＿ＩＦ文に到達すれば（処理６１１）、ｉｆ文の終了部分と関数の終了部分を生成する（処理６１３）。以上のような変換の処理を最終行に到達するまで繰り返す（処理６１４）。

【００３２】以上実行部生成処理（図６の処理６０８～６１４）を図５のルール５１１に対して実行すると、関数５２１のｉｆ文の実行部にあたるブロック５２１ｂ、５２１ｃ、５２１ｄ、５２１ｅが、それぞれルール５１１のブロック５１１ｂ、５１１ｃ、５１１ｄ、５１１ｅから生成され、このうちブロック５２１ｂ、５２１ｃ、５２１ｅは専用言語のコマンド列を対応するＣ言語の標準関数に変換したもの、ブロック５２１ｄはＣ言語によるユーザ関数ブロック５１１ｄの内容をコピーしたものとなっている。また関数５２１の最終部分のブロック５２１ｆは、ルール５１１の最後のＥＮＤ＿ＩＦ文に対応して処理６１３で生成される。図５の続くルール５１２

からも同様にして関数５２２が生成される。

【００３３】以上のように、制御プログラムを汎用言語に変換することにより、汎用言語によるユーザ関数の引用が容易になり、さらにこれらの汎用言語のソースコードをコンパイル、リンクすることで、制御装置において高速実行可能な機械語のオブジェクトコードを生成することが可能となる。

【００３４】なお、本実施の形態では汎用言語としてＣ言語を用いた例を示したが、別の汎用言語（例えば、Ｃ＋＋など）を用いて同様の制御プログラム作成装置を実現することも可能である。また、本実施の形態ではユーザ関数を記述するための汎用言語と専用言語を変換して得られる汎用言語を同じＣ言語とした例を示したが、これをそれぞれ異なった汎用言語（例えば、前者をアセンブリ言語、後者をＣ言語など）とすることも可能である。この場合、ユーザ関数汎用言語ソースコードと制御プログラム汎用言語ソースコードを、それぞれ別の中間機械語変換手段（コンパイラ）により中間機械語オブジェクトコードに変換し、それらを中間機械語オブジェクトコード結合手段により結合することで、同様の制御プログラムの作成方法を実現することが可能である。

【００３５】

【発明の効果】本発明によれば、ユーザが汎用言語ソースコードに含まれる関数の中から必要なユーザ関数を選択するだけで、専用言語の制御プログラムの中にユーザ関数の記述を混在させるための情報が自動的に生成されるので、ユーザ関数を引用するための余計な手間と入力ミスを削減し、プログラム作成効率を向上することが可能となる。さらに、本発明によれば、制御プログラムから制御装置のマイクロプロセッサが直接実行可能なオブジェクトコードが生成されるので、プログラム実行効率を向上することも可能となる。

【図面の簡単な説明】

【図１】本発明になる制御プログラム作成装置の処理手順例を示す図である。

【図２】制御プログラムの編集画面の一例を示す図である。

【図３】ユーザ関数抽出手段による処理の一例を示す。

【図４】ユーザ関数抽出手段の概略処理を示すフローチャートである。

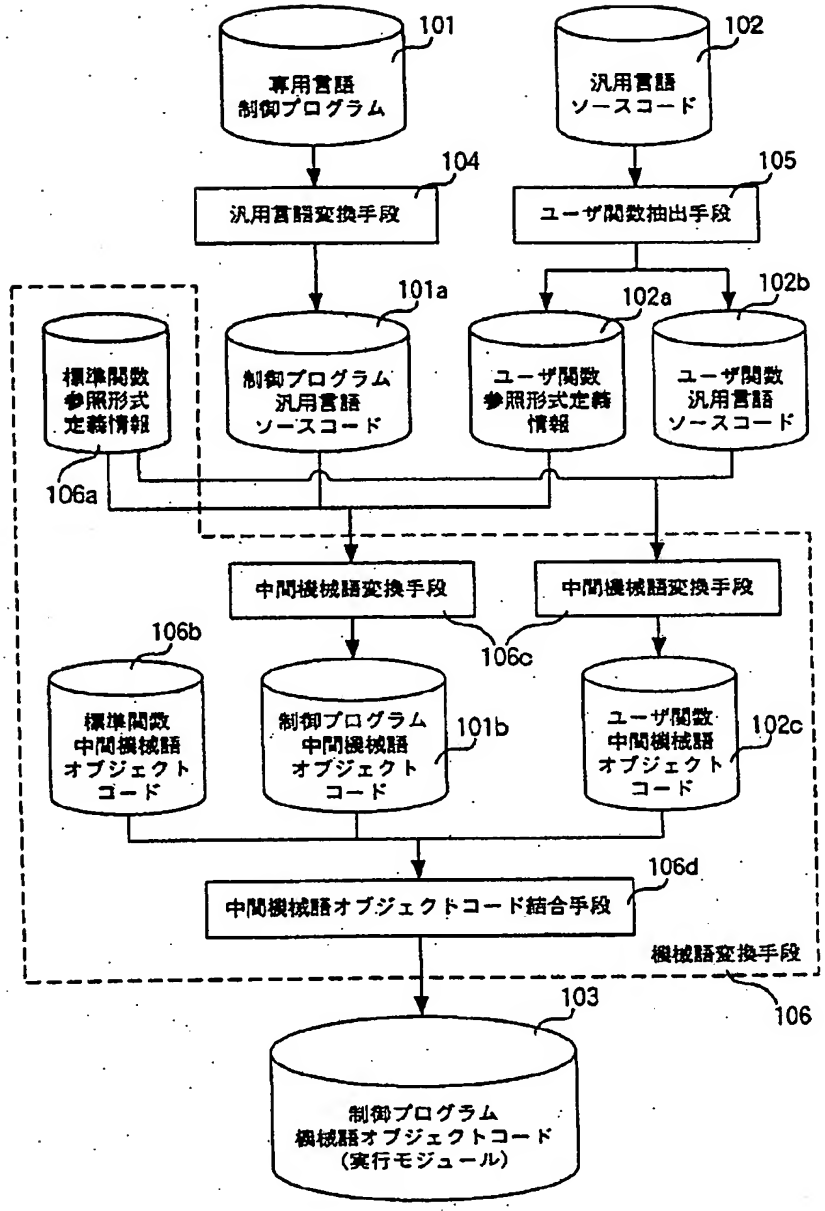
【図５】汎用言語変換手段により、図２に示した専用言語制御プログラムの一部を制御プログラム汎用言語ソースコードに変換した例を示す図である。

【図６】汎用言語変換手段の概略処理を示すフローチャートである。

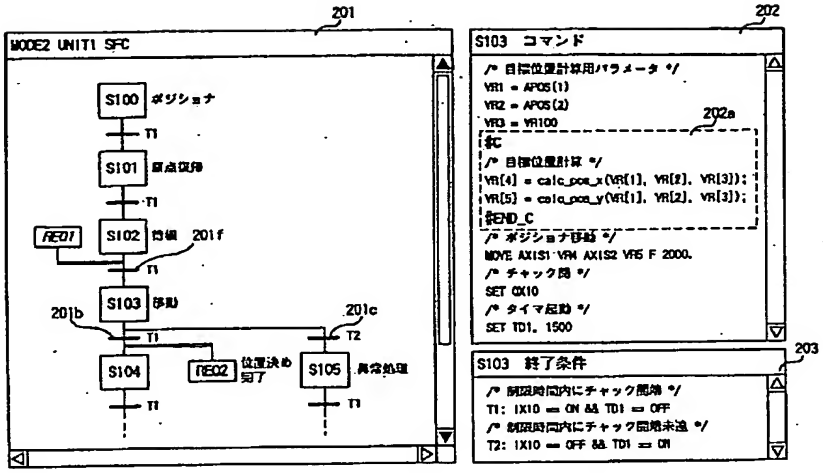
【符号の説明】

１０１　専用言語制御プログラム
１０１ａ　制御プログラム汎用言語ソースコード
１０１ｂ　制御プログラム中間機械語オブジェクトコード
１０２　汎用言語ソースコード
１０２ａ　ユーザ関数参照形式定義情報
１０２ｂ　ユーザ関数汎用言語ソースコード
１０２ｃ　ユーザ関数中間機械語オブジェクトコード
１０３　制御プログラム機械語オブジェクトコード
１０４　汎用言語変換手段
１０５　ユーザ関数抽出手段
１０６　機械語変換手段
１０６ａ　標準関数参照形式定義情報
１０６ｂ　標準関数中間機械語オブジェクトコード
１０６ｃ　中間機械語変換手段
１０６ｄ　中間機械語オブジェクトコード結合手段
２０１　シーケンス編集ウィンドウ
２０２　コマンド編集ウィンドウ
２０２ａ　ユーザ関数ブロック
２０３　終了条件編集ウィンドウ
３０１　Ｃ言語ソースコード
３０２　ユーザ関数のチェックリスト
３０２ａ　チェックボックス
３０２ｂ　ユーザ関数の宣言文
３０３　ユーザ関数の外部参照宣言ヘッダファイル
３０４　ユーザ関数ソースコード
５１０　専用言語制御プログラム
５１１　ルール
５２０　制御プログラム汎用言語ソースコード
５２１　関数

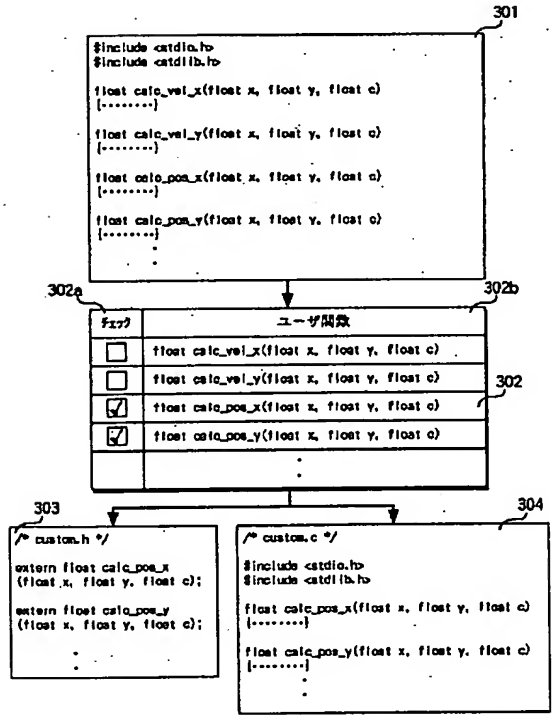
【図１】


101
専用言語
制御プログラム
102
汎用言語
ソースコード
104
汎用言語変換手段
105
ユーザ関数抽出手段
101a
102a
102b
標準関数
参照形式
定義情報
制御プログラム
汎用言語
ソースコード
ユーザ関数
参照形式定義
情報
ユーザ関数
汎用言語
ソースコード
106a
中間機械語変換手段
中間機械語変換手段
106b
106c
標準関数
中間機械語
オブジェクト
コード
制御プログラム
中間機械語
オブジェクト
コード
101b
ユーザ関数
中間機械語
オブジェクト
コード
102c
106d
中間機械語オブジェクトコード結合手段
機械語変換手段
106
103
制御プログラム
機械語オブジェクトコード
(実行モジュール)

【図２】

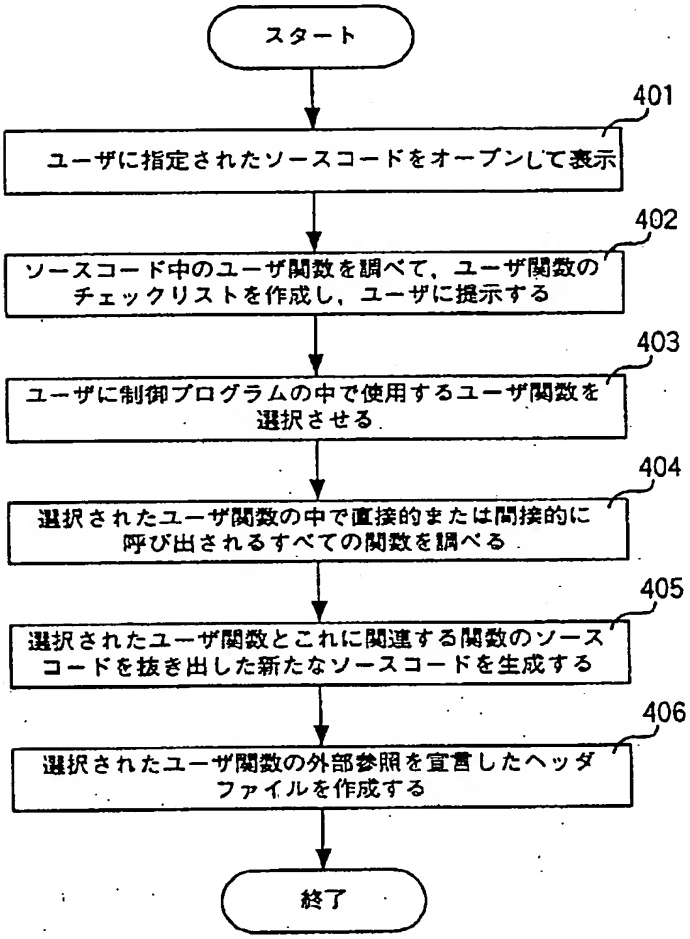

201
MODE2 UNIT1 SFC
S100 ポジショナ
T1
S101 原点復帰
T1
RE01
S102 待機
201f
T1
S103 移動
201b
201c
T1
T2
S104
RE02 位置決め完了
S105 異常処理
T1
T1
202
S103 コマンド
/* 目標位置計算用パラメータ */
VR1 = APOS(1)
VR2 = APOS(2)
VR3 = VR100
202a
#C
/* 目標位置計算 */
VR[4] = calc_pos_x(VR[1], VR[2], VR[3]);
VR[5] = calc_pos_y(VR[1], VR[2], VR[3]);
#END_C
/* ポジショナ移動 */
MOVE AXIS1 VR4 AXIS2 VR5 F 2000.
/* チャック閉 */
SET OX10
/* タイマ起動 */
SET TD1, 1500
203
S103 終了条件
/* 制限時間内にチャック閉確 */
T1: IX10 == ON && TD1 == OFF
/* 制限時間内にチャック閉確未達 */
T2: IX10 == OFF && TD1 == ON

【図３】

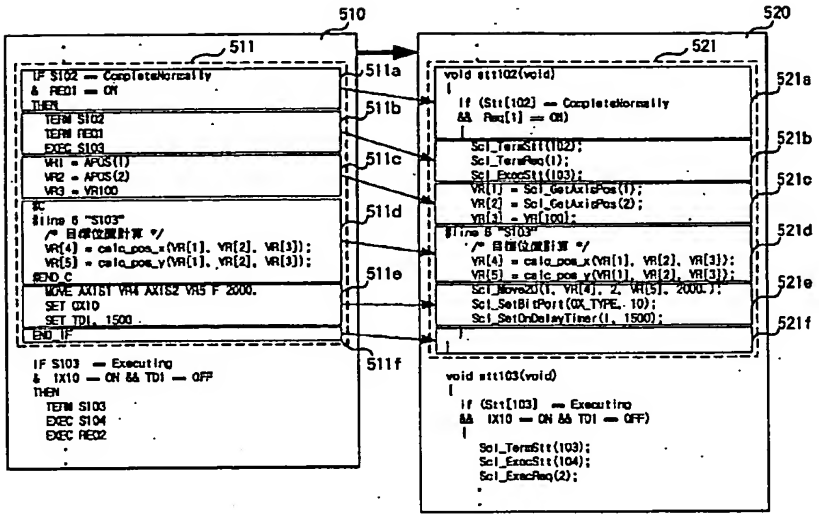

301
$include <stdio.h>
$include <stdlib.h>
float calc_vel_x(float x, float y, float c)
{--------}
float calc_vel_y(float x, float y, float c)
{--------}
float calc_pos_x(float x, float y, float c)
{--------}
float calc_pos_y(float x, float y, float c)
{--------}
302a
302b
チェック
ユーザ関数
float calc_vel_x(float x, float y, float c)
float calc_vel_y(float x, float y, float c)
float calc_pos_x(float x, float y, float c)
float calc_pos_y(float x, float y, float c)
302
303
/* custom.h */
extern float calc_pos_x
(float x, float y, float c);
extern float calc_pos_y
(float x, float y, float c);
304
/* custom.c */
$include <stdio.h>
$include <stdlib.h>
float calc_pos_x(float x, float y, float c)
{--------}
float calc_pos_y(float x, float y, float c)
{--------}

【図４】

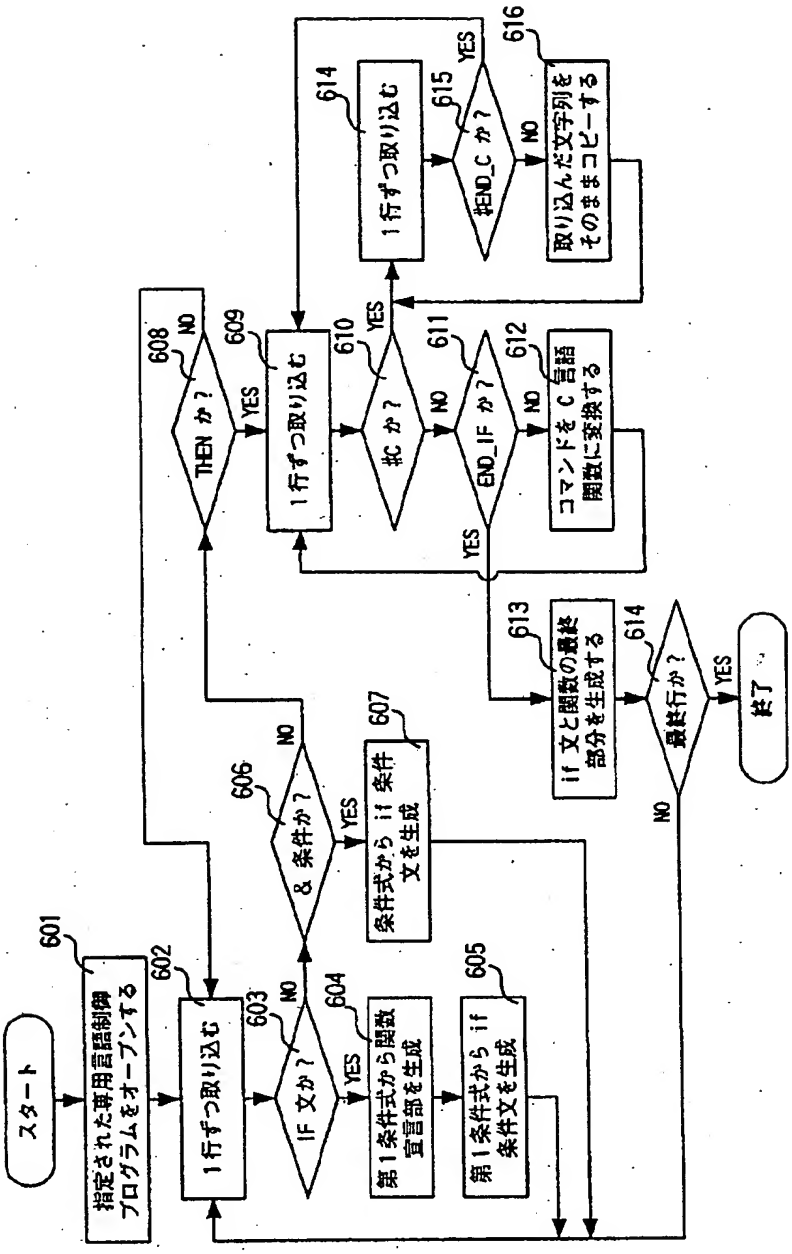
ユーザ関数抽出手段の処理



【図５】


510
511
511a
511b
511c
511d
511e
511f
IF S102 == CompleteNormally
& REQ1 == ON
THEN
TERM S102
TERM REQ1
EXEC S103
VR1 = APOS(1)
VR2 = APOS(2)
VR3 = VR100
$C
$line 6 "S103"
/* 目標位置計算 */
VR[4] = calc_pos_x(VR[1], VR[2], VR[3]);
VR[5] = calc_pos_y(VR[1], VR[2], VR[3]);
$END_C
MOVE AXIS1 VR4 AXIS2 VR5 F 2000.
SET OX10
SET TD1, 1500
END_IF
IF S103 == Executing
& IX10 == ON && TD1 == OFF
THEN
TERM S103
EXEC S104
EXEC REQ2
520
521
521a
521b
521c
521d
521e
521f
void stt102(void)
{
if (Stt[102] == CompleteNormally
&& Req[1] == ON)
{
Scl_TermStt(102);
Scl_TermReq(1);
Scl_ExecStt(103);
VR[1] = Scl_GetAxisPos(1);
VR[2] = Scl_GetAxisPos(2);
VR[3] = VR[100];
$line 8 "S103"
/* 目標位置計算 */
VR[4] = calc_pos_x(VR[1], VR[2], VR[3]);
VR[5] = calc_pos_y(VR[1], VR[2], VR[3]);
Scl_Move2D(1, VR[4], 2, VR[5], 2000.);
Scl_SetBitPort(OX_TYPE, 10);
Scl_SetOnDelayTimer(1, 1500);
}
}
void stt103(void)
{
if (Stt[103] == Executing
&& IX10 == ON && TD1 == OFF)
{
Scl_TermStt(103);
Scl_ExecStt(104);
Scl_ExecReq(2);

【図６】



フロントページの続き

(72)発明者　松下　鶴正
東京都千代田区神田駿河台四丁目６番地
株式会社日立製作所内

(72)発明者　下津　忠夫
東京都千代田区神田駿河台四丁目６番地
株式会社日立製作所内

Ｆターム(参考) 5B081 AA10 CC01
5H220 BB12 CC05 CX02 DD04 DD07
JJ12 JJ24 JJ53